Extension du site de supervision du pilote IPv6 Renater II

Romain Recourt

Romain Recourt

DUT Informatique
2eme année

Extension du site de supervision du pilote IPv6 Renater II

Stage de deuxième année de DUT Informatique

7 avril 2003 – 13 juin 2003

Encadrante : Isabelle Astic
Parrain de stage : Emmanuel Nataf

Romain Recourt

DUT Informatique
2eme année

Extension du site de supervision du pilote IPv6 Renater II

Stage de deuxième année de DUT Informatique

7 avril 2003 – 13 juin 2003

Encadrante : Isabelle Astic
Parrain de stage : Emmanuel Nataf

Remerciements

- Toute l'équipe Madynes, pour son accueil ;
- Isabelle Astic, pour m'avoir suivi pendant ces dix semaines de stage ;
- Emmanuel Nataf, pour ses conseils.

Sommaire

1. Introduction

Le stage présenté ici a été effectué du 7 avril 2003 au 13 juin 2003 au sein de l'équipe de recherche Madynes, au LORIA.

Il s'est déroulé dans le cadre d'un DUT informatique effectué à l'IUT Nancy-Charlemagne durant l'année scolaire 2002-2003.

Après une brève présentation du laboratoire et de l'équipe de recherche Madynes, ce mémoire présentera le contexte de ce stage avant de détailler le travail réalisé. Seront enfin abordés les résultats du stage, et une conclusion sur son intérêt.

2. Présentation de l'entreprise

2.1. Le Loria

2.1.1. Généralités

Le Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA) est une unité mixte de recherche (UMR 7503) commune au CNRS (Centre National de la Recherche Scientifique), à l'INRIA (Institut National de Recherche en Informatique et en Automatique), à l'INPL (Institut National Polytechnique de Lorraine), et aux universités Henri Poincaré Nancy 1 et Nancy 2.

Cette unité, dont la création a été officialisée le 19 décembre 1997 par la signature du contrat quadriennal avec le Ministère de l'Éducation Nationale, de la Recherche et de la Technologie et par une convention entre les cinq partenaires, succède ainsi au Centre de Recherche en Informatique de Nancy (CRIN), et aux équipes communes entre celui-ci et l'Unité de Recherche INRIA de Lorraine.

2.1.2. Les Domaines de recherche

Dans le secteur des sciences et technologies de l'information et de la communication, le LORIA possède, à travers dix neuf équipes de recherche, cent cinquante chercheurs et une centaine de doctorants, des compétences reconnues dans des secteurs en pleine évolution et porteurs de développement économique potentiel. Les activités de ces équipes sont centrées autour de cinq thématiques principales "transversales" sur lesquelles elles développent des recherches fondamentales et appliquées. Bien entendu, des équipes ont des activités dans plusieurs de ces thématiques.

- Calculs, réseaux et graphismes à hautes performances ;
- Télé-opérations et assistants intelligents ;
- Ingénierie des langues, du document et de l'information scientifique et

technique ;
- Qualité et sûreté des logiciels et systèmes informatiques ;
- Bioinformatique et applications à la génomique.

2.2. L'équipe MADYNES

Cette équipe, Managing Dynamic Networks and Services (gestion de réseaux et de services dynamiques) a été créée à la suite du projet RESEDAS. Le projet MADYNES vise "la conception, la validation et la mise en oeuvre de nouveaux paradigmes et architectures de supervision et de contrôle capables de maîtriser la dynamicité croissante des infrastructures et services de télécommunications et de résister au facteur d'échelle induit par l'Internet ubiquitaire".

Le projet s'articule donc autour de deux domaines de recherche principaux :

- Gestion autonome :
    - Elaboration de méthodes d'auto-organisation des entités de gestion ;
    - Conception, évaluation et mise en oeuvre d'architectures de supervision exploitant le modèle Pair-à-Pair (P2P), le routage applicatif, et de nouvelles approches pour la représentation de l'information de gestion ;
    - Modélisation et évaluation des performances des infrastructures de supervision .
- Aires fonctionnelles :
    - Sécurité : nouveaux protocoles de distribution de clés et infrastructures pour le respect de l'anonymat et de la vie privée ;
    - Configuration et provision de services ;
    - Mesure, analyse et instrumentation automatique des services.

Le principal domaine d'application des résultats de ces axes de recherche est Internet nouvelle génération (IPv6), son architecture et les services associés présentant à la fois dynamicité et besoin de passage à l'échelle abordés dans les autres axes du projet. Ceux-ci peuvent être représentés par la "fleur" suivante.

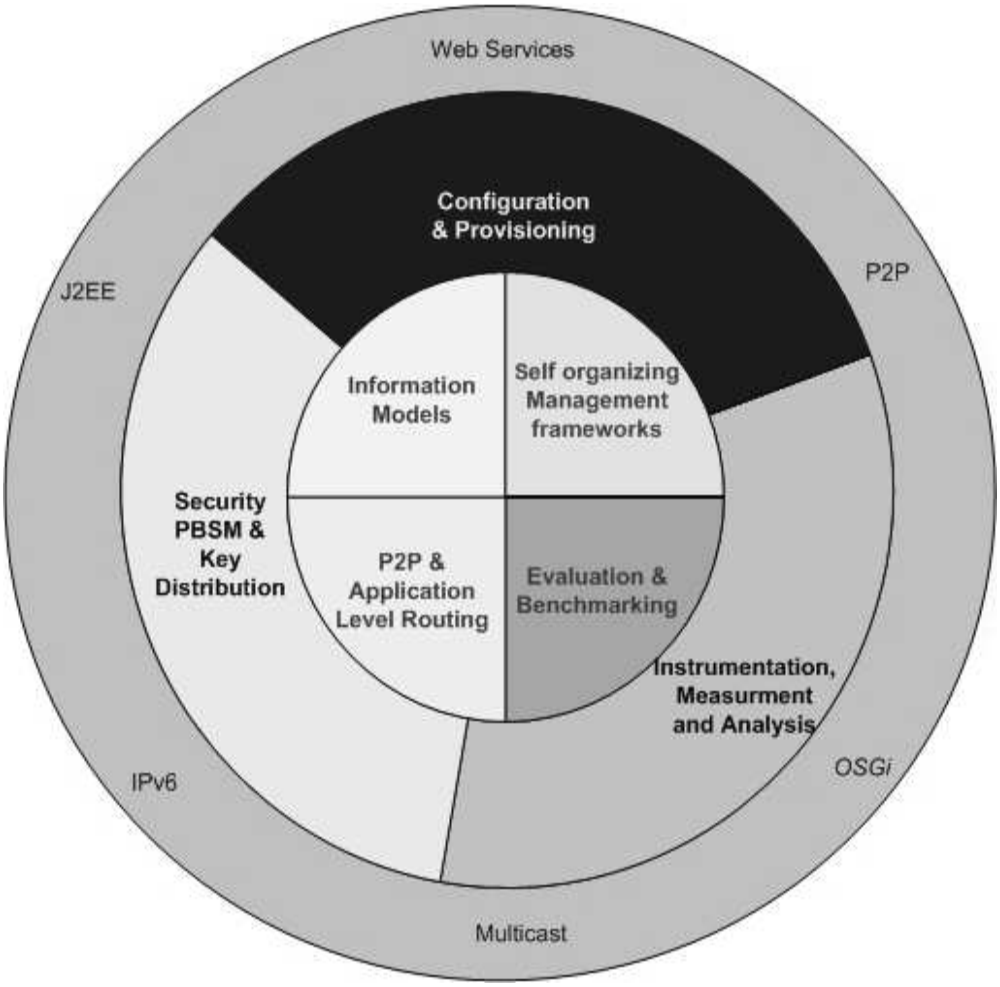


fig 1 : Axes de recherche de l'équipe Madynes

3. Contexte

3.1. IPv6

IPv6 est la nouvelle version d'IP (Internet Protocol) devant se substituer à l'actuel protocole utilisé (IPv4). Sa principale caractéristique est un espace d'adressage étendu grace à des adresses codées sur 128 bits (au lieu de 32bits pour IPv4). Une adresse IPv6 se note de la façon suivante : on la décompose en 8 groupes de 4 chiffres hexadécimaux (soit 8 fois 16 bits) séparés par des deux-points. Par exemple :

```
8000:0000:0000:0000:0123:4567:89AB:CDEF
```

On peut aussi trouver des adresses comprennant deux fois le symbole deux-points consécutivement. Il s'agit d'une optimisation de notation, puisque les adresses contiennent souvent des séquences de zéro assez longues. On a le droit de remplacer une séquence de 16 zéro ou plus (en héxadécimal, soit 64 bits consécutifs à zéro) par deux symboles deux-points. Ainsi l'adresse précédente peut se noter :

```
8::123:4567:89AB:CDEF
```

3.2. Le Pilote IPv6 Renater II

Le Pilote IPv6 Renater II est une épine dorsale du réseau français de test pour IPv6. Il rassemble plusieurs plate-formes de test, raccordées entre elles via des Points d'Interconnexion Organisationnels (PIO) et des Points d'Interconnexion Régionaux (PIR). De plus, elles sont toutes reliées au Noeud d'Interconnexion d'Opérateur (NIO) de Paris qui permet de relier les pilote IPv6 Renater II à d’autres réseaux IPv6. Le pilote IPv6 Renater II a la structure suivante (cf fig. 2).

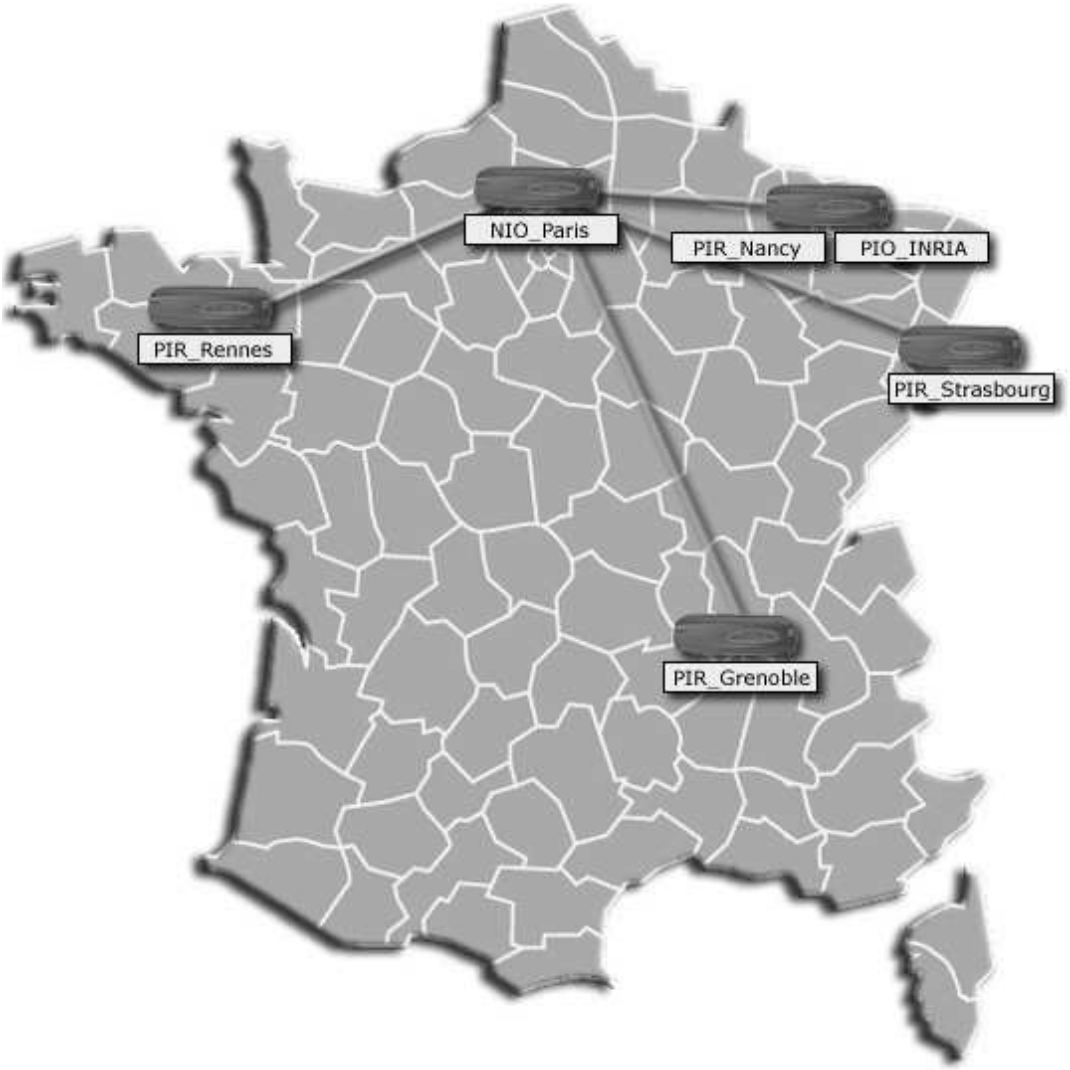


fig 2 : Structure du pilote IPv6 Renater II

3.3. Supervision du réseau

3.3.1. Le site de supervision : w6.loria.fr

Il existe actuellement au niveau du pilote IPv6 Renater II un site de supervision (http://w6.loria.fr) permettant d'obtenir des informations sur l'état du réseau et de ses routeurs en particulier. Depuis ce site, l'utilisateur peut obtenir des informations de deux natures sur un routeur de son choix. Il a d'abord accès à des données concernant le trafic au niveau de chaque routeur sous forme de graphiques générés à l'aide de RRDTool. Il peut choisir entre le trafic IPv6 global, l'ICMP ou l'UDP. Voici un exemple de statistiques IPv6 sur le PIR de Nancy (cf figure 3). A gauche, nous avons le trafic moyen en nombre de paquets par seconde et à droite le cumul des paquets traités.

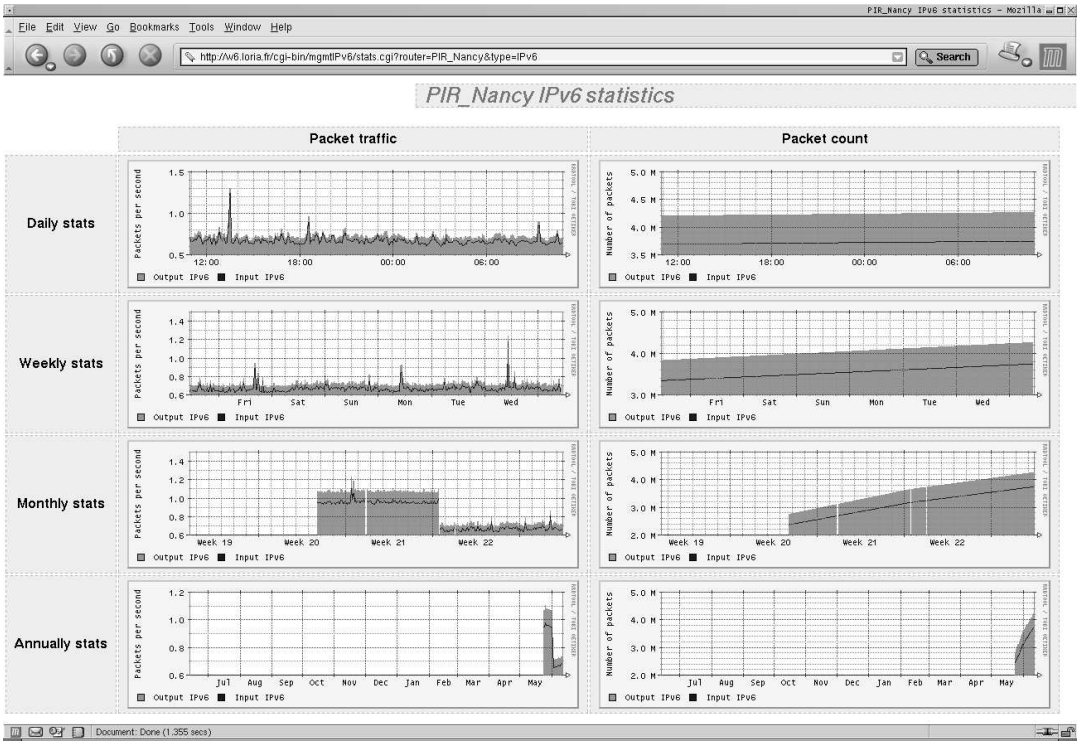


fig 3 : Exemple de graphique statistique de frafic sur un routeur

Il y a aussi possibilité d'interroger directement le routeur pour obtenir des informations plus détaillées. Pour cela, le site met à disposition un outil, le Looking Glass, qui permet d'exécuter des commandes de demande d'informations sur un routeur (tables de routage, trafic, voisinage, entre autres) et d'afficher le résultat dans une page web.

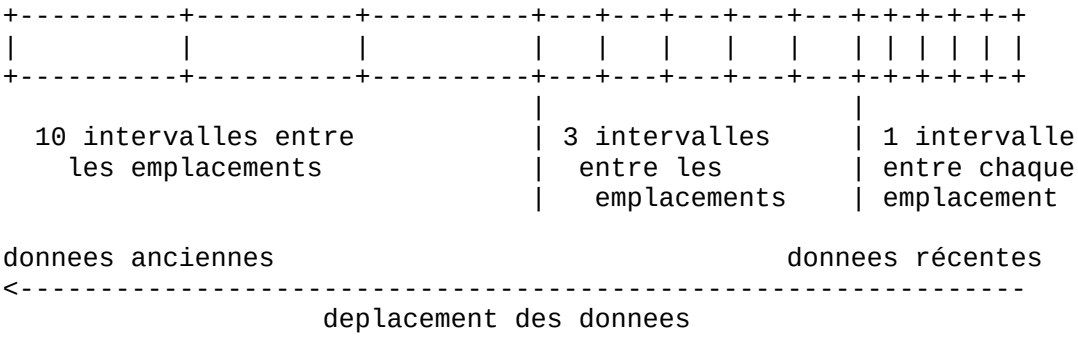
L'accès aux informations sur chaque routeur se fait via une carte de France des routeurs (cf. Fig. 2 en 3.2.) qui ne fournit aucune autre information que la localisation géographique des routeurs.

3.3.2. Les outils utilisés

3.3.2.1. RRDTool

RRDTool (Round Robin Database Tool) est un programme écrit par Tobias Oetiker permettant de collecter, traiter et synthétiser des informations d'état en fonction du temps. Le principe est de stocker les informations fournies dans un espace de taille fixe, en dégradant la précision des données les plus anciennes. Ainsi, des données datant de quelques heures pourront avoir une précision des quelques minutes, tandis que des données vieilles de plusieurs années n'auront une précision que de quelques jours. l'idée de base est simple : on défini un intervalle de temps de base (par défaut, 300 secondes, soit 5 minutes) puis des niveaux de précision avec le nombre de données associées. Les données entrées sont affectées à l'emplacement dont la date (à la seconde près) correspond le mieux à la date fournie pour les

données.



Les principales caractéristiques de RRDTool sont :

- La taille fixe de la base de données : les emplacements disponibles sont créés au début et ne peuvent être modifiés par la suite ;
- L'outil de génération de graphiques, très puissant, qui permet de visualiser simplement les données (cf figure x);
- La consolidation des données (perte de précision) qui ne permet pas de tenir d'historique précis des données.

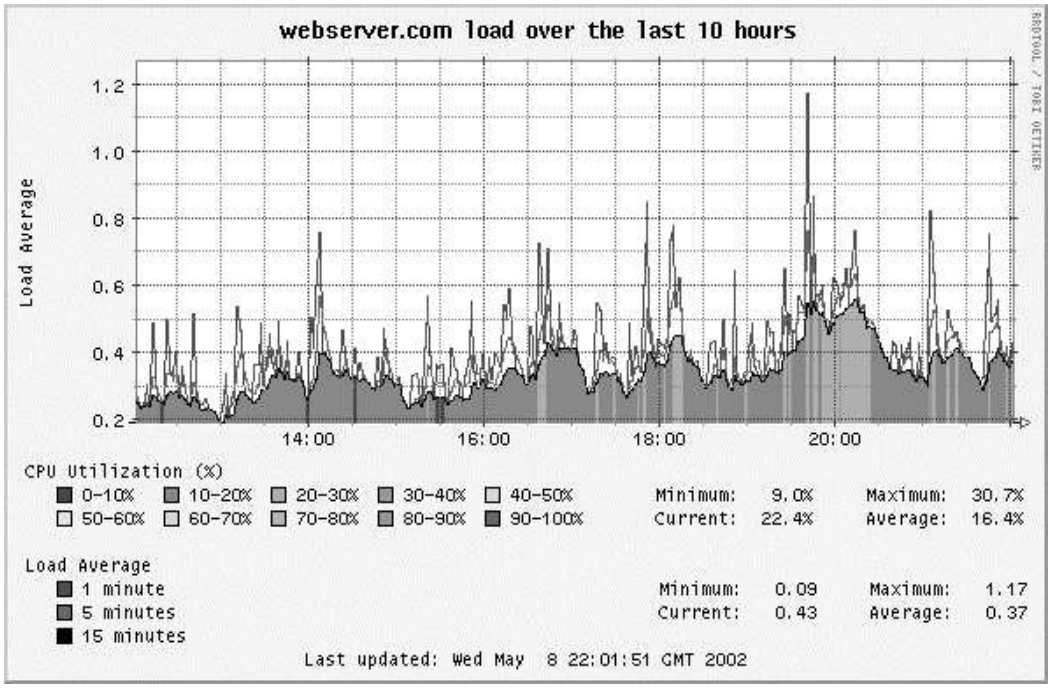


fig 4 : Exemple d'utilisation de RRDTool

3.3.2.2. Looking Glass

Le Looking Glass est un outil écrit en PERL permettant de récupérer des informations sur un routeur à l'aide d'un navigateur Internet. Cet outil existait pour IPv4 et a été modifié au Loria pour fonctionner avec IPv6. Il permet de consulter

les tables BGP (Border Gateway Protocol), d'obtenir des informations sur le trafic, les interfaces, les routeurs voisins, mais aussi d'executer des commandes de test du réseau, comme ping ou traceroute. Sous sa forme actuelle, le Looking Glass est capable de récupérer toutes ces informations sur des routeurs Cisco et Juniper de manière tout à fait transparente pour l'utilisateur.

Une consultation à l'aide du Looking Glass revient à executer les commandes de requête d'informations sur un terminal connecté au routeur. Voici par exemple le formulaire du Looking Glass rempli de façon à récupérer les tables de routage du PIO INRIA (cf fig. 5).

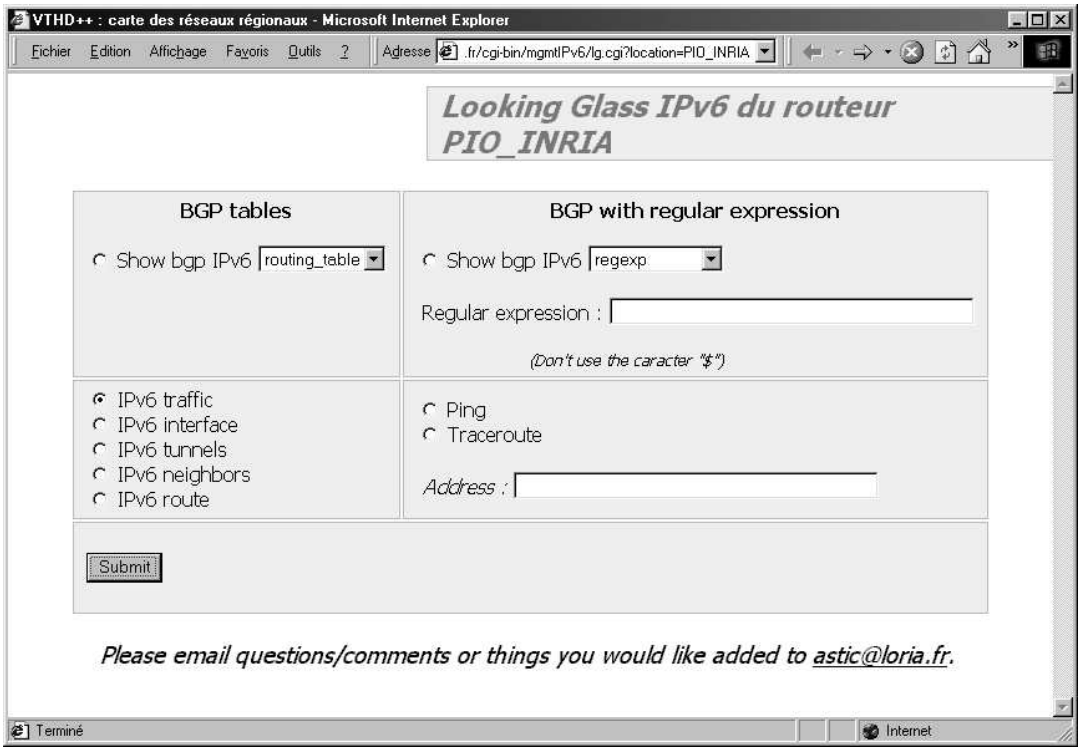


fig 5 : Interface du Looking Glass

Et voici le résultat (fig. 6):

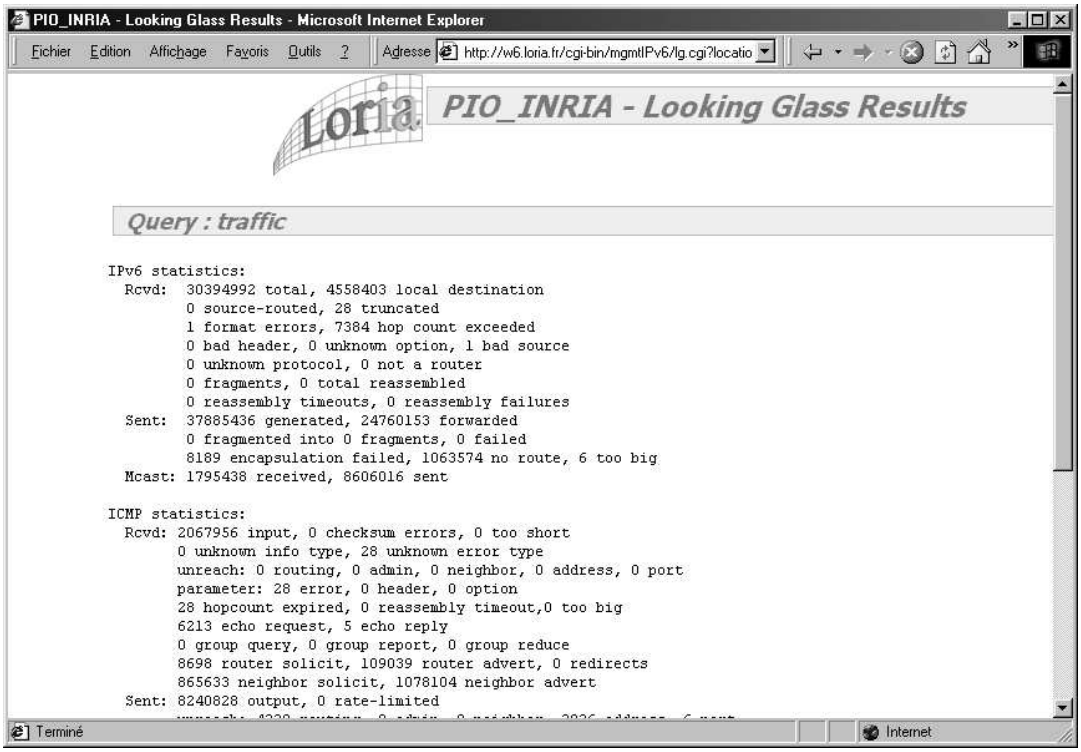


fig 6 : Résultat d'une demande d'informations à l'aide du Looking Glass

On constate qu'il n'y a aucune mise en forme sur les données récupérées, si ce n'est leur intégration dans une page HTML. De plus, comme la requête est effectivement envoyée au routeur, les informations fournies le sont en temps réel.

3.4. Objectifs du stage

Les objectifs de ce stage étaient les suivants :

- Fournir des informations à l'utilisateur dès l'apparition de la carte ;
- Améliorer les performances, en particulier lors de l'accès aux graphiques ;
- Rechercher un maximum d'optimisations possibles et les appliquer (objectif secondaire).

4. Déroulement du stage

4.1. Analyse et conception

4.1.1. Etude Initiale

La première partie de mon travail d'analyse a consisté en l'étude des objectifs, afin de déterminer la meilleure façon de les réaliser. Tout d'abord, les informations à ajouter au niveau de la carte sont assez limitées, et l'une des informations les plus utiles est l'état du routeur (répond ou ne répond pas). Il existe alors plusieurs méthodes pour faire apparaître ce genre d'information :

- Découper l'image, placer les morceaux dans un tableau HTML et changer les parties contenant les images des routeurs en fonction de leur état ;
- Placer un fond de carte statique et placer dynamiquement les images des routeurs au-dessus, en utilisant le DHTML ;
- Générer une image à partir du fond de carte et des images des routeurs.

La première solution n'est pas viable, puisque ralentissant beaucoup l'affichage et ne permettant plus la moindre modification (ajout ou suppression de routeur). La deuxième serait idéale si tous les navigateurs savaient interpréter correctement le DHTML, ce qui est loin d'être le cas. La dernière nécessite un traitement plus important et coûteux en temps machine (manipulation d'images). La compatibilité avec tous les navigateurs étant une contrainte pour ce site, et l'accélération des temps de réponses étant un autre objectif principal, la troisième solution a été adoptée.

Restait à voir de quelle façon et surtout à quel moment serait générée cette carte. Une solution serait de la créer par un script CGI au moment de la consultation, mais le temps de réponse serait conséquent. Une autre solution serait de faire générer cette carte par un programme tournant en tache de fond, un démon, qui se chargerait de créer une nouvelle carte toutes les cinq minutes (l'intervalle entre deux communications des démons de collecte d'informations avec les routeurs), ce qui semble beaucoup plus raisonnable, puisque la puissance machine nécessaire

serait indépendante du nombre de consultations.

Sur ce point, la solution choisie est donc un démon générant toutes les cinq minutes une image représentant une carte de France avec les routeurs testés représentés différemment selon qu'ils répondent ou non.

L'accélération ensuite. La principale cause de ralentissement est l'utilisation de scripts CGI pour générer les graphiques, ce qui implique d'effectuer les traitements nécessaires à la visualitation des graphiques à la consultation.La solution retenue était de générer ces graphiques à part, par un démon qui effectuerait tous les traitements nécessaires toutes les cinq minutes (à chaque arrivée de données fraiches). La consommation en terme de temps machine serait ainsi importante, mais constante : il n'y aurait plus à craindre une saturation de capacité, et donc des délais encore plus longs en cas de nombreuses consultations simultanées.

La deuxième partie de cette tâche a consisté en l'observation du code des scripts de génération des pages HTML et des graphiques, à la recherche d'améliorations possibles. Tout d'abord, je me suis documenté sur les outils utilisés (en particulier RRDTool) ainsi que sur le langage mis en oeuvre : PERL. Ensuite, j'ai débuté l'exploration du code existant afin d'en comprendre le fonctionnement. Une fois celui-ci assimilé, j'ai décrit les points à améliorer dans le code lui-même, l'architecture globale du site et des scripts le composant, ainsi que du rendu au niveau de l'interface utilisateur. J'ai ainsi relevé de nombreux points sujets à amélioration, en plus des modifications imposées par les objectifs fixés. En voici quelques exemples :

- Non conformité de certaines pages HTML malgré une certification XHTML 1.0 ;
- Unités erronées au niveau de certains graphiques ;
- Artefacts sur les graphiques de cumul donnant l'impression d'une baisse de la courbe de cumul (erreurs d'arrondis à la génération) ;
- Quelques liens morts sur le site ;
- Manque de cohérence dans les références de ressources au sein du site : cohabitation d'adresses absolues, relatives et d'alias serveur.
- Informations de configuration codées en dur dans différents scripts, d'où redondance des données et complication de la maintenance ;
- Scripts redondants : présence de 96 scripts quasi-identiques différant seulement de quelques valeurs pouvant être passées en paramètres (scripts de génération des graphiques) ;
- Incohérence entre certaines références et l'arborescence des fichiers ;
- Structure figée, impossibilité d'ajouter un nouveau routeur ;
- Indépendance totale du Looking Glass et de ses données, entraînant une redondance d'informations supplémentaire.

Une fois cette liste des modifications souhaitables établie, il m'a fallu faire le lien entre les modifications de l'interface et celles liées à la structure du code. Ensuite, j'ai cherché des solutions techniques permettant d'apporter un maximum de modifications dans le temps imparti, tout en tenant compte des priorités imposées. Le plus délicat étant de définir en parallèle un ordre de traitement qui permettrait à chaque changement de se baser sur les précédents sans pour autant être annulé par les suivants.

Une fois la structure du site connue, les objectifs définis et les priorités établies, il était possible d'avoir une vision d'ensemble du résultat escompté, et les différences par rapport à l'existant (cf fig 7 et 8).

Le principe général est de limiter les traitements en temps réel, c'est à dire effectués au moment de la consultation au profit de taches s'executant en fond de manière asynchrone, qui seraient chargées de générer des données non-dynamiques régulièrement remises à jour. Ainsi, au prix d'un décalage minime (écart de temps entre deux générations des ressources) on obtient un gain de performances appréciable, ainsi qu'une plus grande tolérance à la charge (les connexions simultanées n'entrainent plus une quantité de traitement plus importante)

De plus, tout le site étant en PERL (démons, CGI et autres outils), il était nécessaire de conserver ce langage afin de pouvoir réutiliser les outils existants. La meilleure solution semblait donc être de reprendre les éléments existants et d'appliquer les améliorations une à une dans un ordre permettant à chaque fois de tester en conditions réelles chaque amélioration.

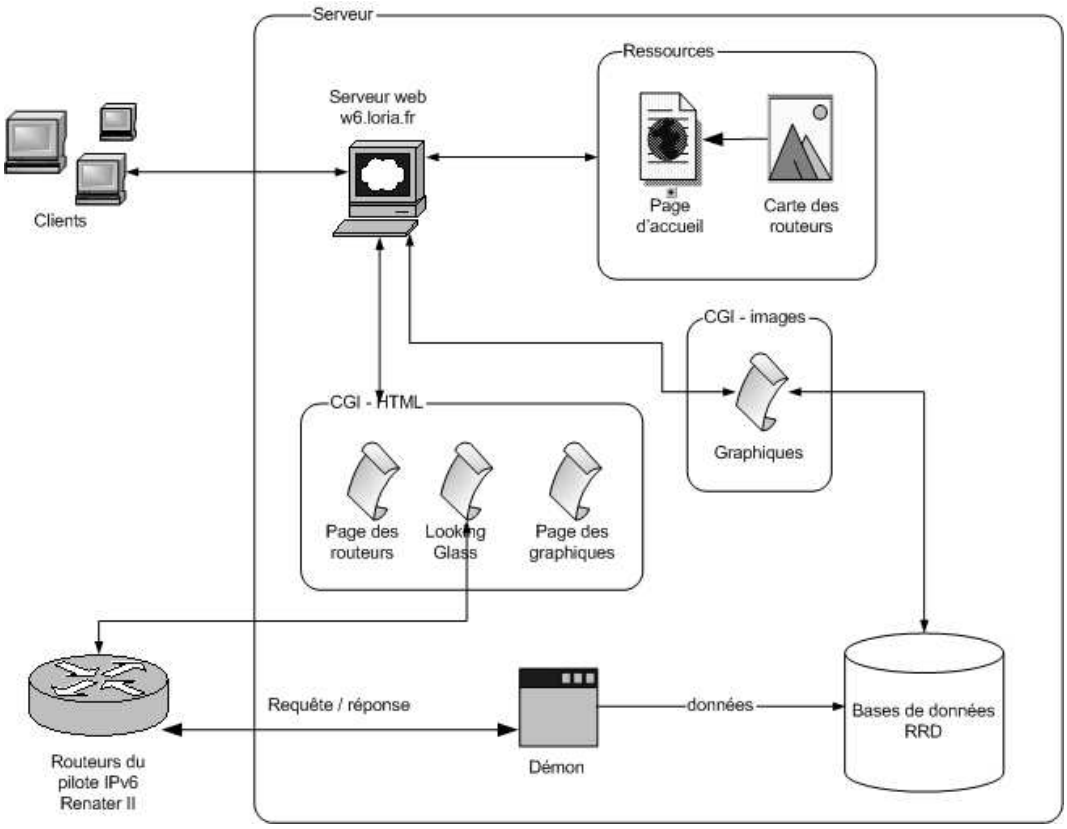


fig 7 : Architecture du site de supervision avant modification

.

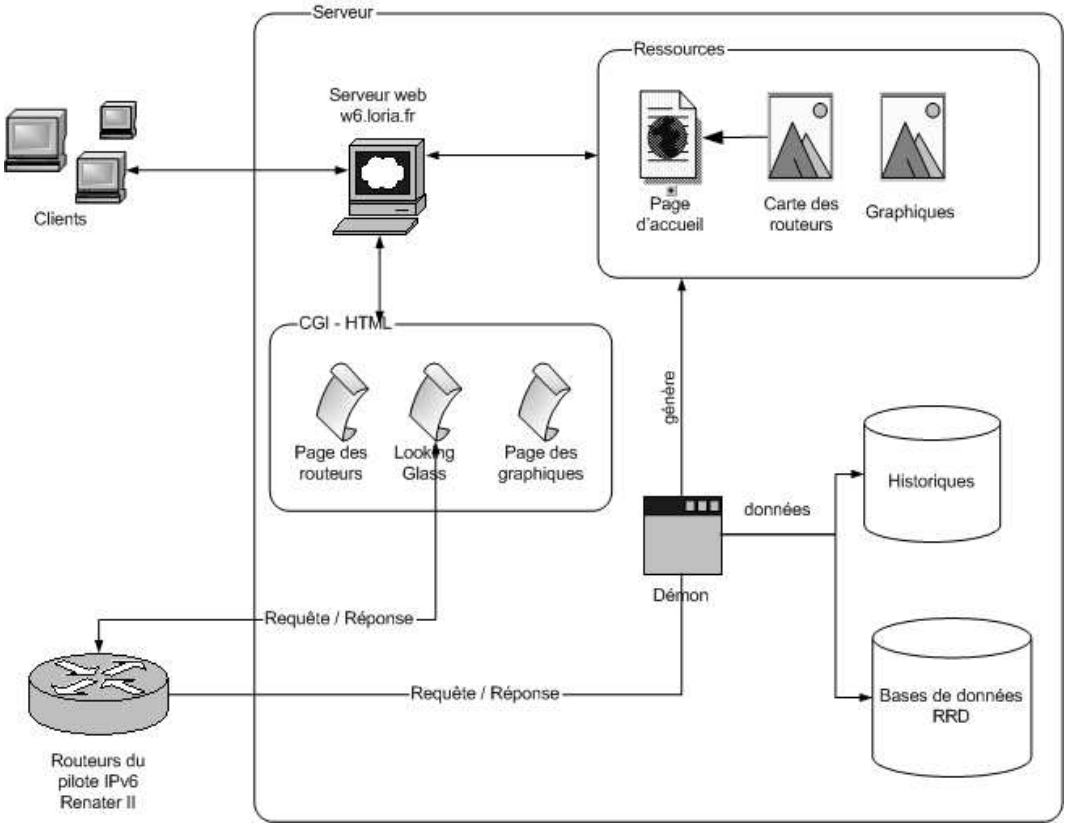


fig 8 : Architecture du site de supervision visée après amélioration.

4.1.2. Planning prévisionnel

Ce planning a été établi pendant le début de la phase d'analyse et ajusté en fonction des résultats de celle-ci. Ainsi ce planning n'a-t-il été complètement arrêté qu'après deux semaines.

1. Etude de l'interface (1 semaine)
    - Familiarisation, prise en main, utilisation du site ;
    - Recherche de points à améliorer.
2. Etude des programmes (2 semaines)
    - Familiarisation, compréhension ;
    - Documentation annexe (RRDTool, CGI, PERL) ;
    - Recherche des points à améliorer (y compris modifications imposées) ;
    - Recherche des solutions pour la partie étude de l'interface ;
    - Mise en relation des solutions pour établir les priorités tout en restant cohérent.
3. Réalisation (3 semaines)
    - Application des modifications dans l'ordre logique ;
    - Tests ;
    - Présentation du travail réalisé.
4. Finalisation (3 semaines)
    - Tests supplémentaires ;
    - Application de corrections et améliorations mineures ;
    - Intégration, remplacement de l'existant ;
    - Recherche d'autres améliorations potentielles.

4.2. Réalisation

4.2.1. Carte dynamique des routeurs

Sur cette partie, la plus importante du stage, je me suis très vite retrouvé face à un problème de taille : pour générer cette carte dynamiquement, il me fallait combiner des données récupérées par 4 programmes différents tournant en parallèle. La solution la plus simple était de stocker ces données dans un ou plusieurs fichiers texte pour ensuite les récupérer à la génération de la carte, mais je me suis retrouvé confronté à des problèmes de concurrence d'accès et de désynchronisation des données. Pour résoudre ce problème, il me fallait d'abord grouper tous les programmes me fournissant les informations souhaitées (les démons chargés de récupérer les informations sur les routeurs) en un seul programme dans lequel je pourrais intégrer la génération de la carte. J'ai donc fourni une solution provisoire basée sur des fichiers texte en attendant d'avoir apporté les modifications nécessaires sur les démons avant de conclure cette partie. Au final, la carte est générée par le démon en utilisant directement les données recueillies, sans avoir à les faire transiter par un fichier externe.

L'idée de base dans la réalisation de cette carte était de superposer sur un fond de carte des images représentant les routeurs répondant ou ne répondant pas. Pour cela, il m'a fallu trouver un outil graphique permettant de générer une image à partir d'un script PERL en superposant plusieurs images avec transparence. Il me fallait donc un outil sachant gérer un des deux formats graphiques courants utilisant la transparence : le GIF et le PNG. Malheureusement, le format GIF étant breveté, il devient difficile de trouver des outils libres utilisant le GIF. J'ai fini par choisir la GDlib, de Thomas Boutell. Elle permet la majorité des actions disponibles sur un éditeur graphique simple : dessin de formes géométriques, surimpression d'images, gestion de la transparence, ainsi que l'ajout de texte. De plus, cette librairie dispose de nombreuses interfaces en PERL, PHP, TCL et de nombreux autres langages.

Au terme de cette partie de mon travail, j'avais un script à intégrer au démon générant une carte au format PNG sur laquelle les routeurs s'affichent d'une couleur différente selon qu'ils répondent, ne répondent pas, ou ne sont pas interrogés. De plus, le nom de chaque routeur venait s'incruster sur la carte, même si cette fonctionnalité n'est pas disponible dans le contexte, la configuration du serveur ne permettant pas cette opération.

Des informations supplémentaires (état des routeurs) sont apportées sur la carte, cet objectif est donc rempli.

4.2.2. Amélioration des performances

Pour cette deuxième partie, ma première tâche a été de simplifier l'arborescence existante : en effet, 96 scripts différents servaient à générer les graphiques, soit un par graphique par routeur. J'ai donc ajouté des paramètres me permettant de ne plus utiliser qu'un seul script CGI pour générer tous les graphiques.

Ensuite, j'ai modifié ce script en supprimant son typage CGI pour l'utiliser dans le démon chargé de récolter les informations sur les routeurs et de générer la carte.

Une fois cette modification appliquée, on constate un gain de temps non négligeable : Une page contenant 8 graphiques, qui auparavant mettait en moyenne 6 secondes à s'afficher ne met plus qu'une seconde à charger complètement.

Cette amélioration apporte un gain de performances de 83% sur la ressource la plus coûteuse en termes de temps de traitement. L'objectif est donc rempli.

4.2.3. Autres améliorations et extensions

La plus importante des améliorations apportées en ce qui concerne la qualité du site est la synthèse de toutes les données de configuration dispersées dans les scripts en un seul fichier de configuration global. Ainsi, l'ajout ou la suppression d'un routeur à la liste se fait simplement, tandis que la redondance et la dispersion des données rendaient cette tache ardue auparavant.

Dans la suite de cette optimisation, on peut noter la simplification de l'arborescence, et le groupement de tous les traitements en un seul démon, script tournant en boucle chargé de récupérer les données sur les routeurs et les stocker, de générer la carte de France et les graphiques. Ce sscript effectue aussi les traitements ajoutés par la suite, dans le cadre des modifications suivantes.Au final, environ 200 fichiers ont été groupés en trois fichiers : un script d'initialisation (création des bases de données), un démon effectuant tous les traitements nécessaires, et un fichier de configuration global. La maintenance future en est encore une fois grandement facilitée.

Par la suite, j'ai aussi implanté la gestion d'un historique précis, sous la forme d'un fichier texte pour chaque routeur contenant les données datées récupérées sur celui-ci. Il n'a pour le moment aucune application pratique, mais il permettra dans le futur de proposer à l'utilisateur de consulter des données anciennes.

A ces extensions assez importantes viennent s'ajouter diverses corrections, améliorations et mises aux normes. Globalement, le site s'en trouve plus fiable, plus simple à maintenir et plus léger (2600 lignes de code redondant supprimées). J'ai donc rempli en grande partie l'objectif secondaire.

5. Conclusion

5.1. Bilan

Après ces dix semaines de stage, je constate que les objectifs primaires sont remplis, que les objectifs secondaires le sont en bonne partie, et que le planning a été respecté à quelques jours près, rattrappés dans la suite du stage. Au niveau réalisation, ce stage est donc une réussite.

5.2. Perspectives

Même si les objectifs du stage ont été remplis, certaines modifications suggérées n'ont pas pu être appliquées. De plus, la nouvelle structure simplifiée permet d'envisager des amoliorationss beaucoup plus lourdes.

- Tout d'abord, certaines données dynamiques sont toujours inscrites dans des fichiers de configuration sans mise à jour, comme l'AS. Il faudrait donc interroger les équipements régulièrement pour récupérer ces données.

- Les données sont maintenant sauvegardées sans perte de qualité, mais il n'y a pas d'interface utilisateur permettant de visualiser les informations sur une période particulière

- Le Looking Glass est toujours une entité indépendante avec ses propres fichiers de configuration, il serait souhaitable de reprendre la structure existante et de finir la mise en commun des fichiers de configuration, déja préparée par la structure commune des fichiers de configuration.

- Il est pour le moment impossible de contrôler d'autres équipements que des routeurs Cisco, et même au niveau de ceux-ci, il existe des problèmes de configurations et de modèles différents qui limitent la dynamicité de la configuration des outils.

- Il serait aussi possible de collecter d'autres informations sur les équipements, afin de fournir plus de renseignement sous forme visuelle ; par exemple en affichant l'état des connexions entre les routeurs sur la carte.

5.3. Apports personnels

Ce stage m'a permi de gérer un projet depuis sa phase d'étude jusqu'à l'intégration, ce qui est pour moi une expérience très enrichissante, puisque couvrant tous les domaines d'attribution d'un analyste-programmeur.

De plus, j'ai eu la chance d'évoluer au sein d'un laboratoire de recherche en informatique, ce qui m'a donné un aperçu d'un autre aspect de l'informatique : le monde de la recherche.

Enfin, j'ai pu pendant ces dix semaines m'intéresser à des technologies et systèmes qui n'ont pas été étudiés dans le cadre du DUT, me donnant ainsi des connaissances complémentaires à celles reçues de l'enseignement de DUT.

Glossaire

BGP : Border Gateway Protocol

CGI : Common Gateway Interface

Démon : Programme fonctionnant en tache de fond, sans interraction avec l'utilisateur

DHTML : Dynamic HyperText Markup Language

GIF : Graphical Interchange Format

HTML : HyperText Markup Language

ICMP : Internet Message Control Protocol

INRIA : Institut National de Recherche en Informatique et Automatisme

IP : Internet Protocol

Loria : Laboratoire Lorrain de Recherche en Informatique et Applications

Madynes : Managing Dynamic Networks and Services

NIO : Noeud d'Interconnexion d'Opérateur

PERL : Practical Extraction and Report Language

PIO : Point d'Interconnexion Organisationnel

PIR : Point d'Interconnexion Régional

PNG : Portable Network Graphics

RRD : Round Robin Database

Script : Programme interprété

TCP : Transmission Control Protocol

UDP : User Datagram Packet

XHTML : eXtended HyperText Markup Language

Bibliographie

http://w6.loria.fr : Site de supervision du pilote IPv6 Renater II.

http://www.boutell.com/gd/ : Site officiel de la GDlib.

http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/ : Site officiel de RRDTool.